台科大 **111** 學年度「彩色影像處理」 作業二：影像分析應用

M11125016 蕭強

Programming language: Python

第一部分「景深擴張」

```python
import numpy as np
import cv2

__lapFil__ = np.array([[-1, -1, -1],
                       [-1, 8, -1],
                       [-1, -1, -1]])


def imgProc(img):
    """
    讀取影像
    """
    img = img / 255
    img = img.astype(np.float32)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    return img


def laplacian_filter(img):
    """
    使用 Laplacian 濾鏡進行影像銳利化
    """
    img = imgProc(img)
    h, w = img.shape
    kh, kw = __lapFil__.shape
    pad = int((kh - 1) / 2)
    pad_img = np.pad(img, (pad, pad), "symmetric")
    output = np.zeros([h, w], dtype=np.float32)
    for y in range(h):
        for x in range(w):
            ci = pad_img[y:y+3, x:x+3]  # ci is crop image
            result = (ci * __lapFil__).sum()
            output[y, x] = result
    return abs(output) * 3
```

```python
def median_filter(img):
    """
    均值綠波濾鏡使影像不破碎化
    """
    mean_kernel = np.ones([23, 23]) / 23**23
    h, w = img.shape
    kh, kw = mean_kernel.shape
    pad = int((kh - 1) / 2)
    pad_img = np.pad(img, (pad, pad), "symmetric")
    output = np.zeros([h, w], dtype=np.float32)
    for y in range(h):
        for x in range(w):
            cp = pad_img[y:y+23, x:x+23]
            result = (cp * mean_kernel).sum()
            output[y, x] = result
    return output


def imgTreshold(img):
    """
    二值化函數
    """
    h, w = img.shape
    output = np.zeros([h, w], dtype=np.float32)
    for y in range(h):
        for x in range(w):
            pixel = img[y, x]
            if pixel > 0:
                pixel = 1
            else:
                pixel = 0
            output[y, x] = pixel
    return output


def cvtDtype(img):
    output = (img * 255).astype(np.uint8)
```

```
    return output


if __name__ == "__main__":
    # 2. 讀取並顯示對焦在前景(fg)與背景(bg)的兩幅影像，並轉換至 float 格式。
    fgimg = cv2.imread('./depthOfField/2fg.jpg')
    bgimg = cv2.imread('./depthOfField/2bg.jpg')
    # 4. 高通濾波：將兩幅影像由彩色轉換至灰階格式，並分別做 Laplacian 高通濾波後，取絕對值。
    fg_hipass = laplacian_filter(fgimg)
    bg_hipass = laplacian_filter(bgimg)
    # 6. 製作前景遮罩 mask = fg_hipass - bg_hipass
    mask = fg_hipass - bg_hipass
    # 7. 將遮罩做「均值濾波」，濾鏡尺寸要很大，才不至於使區塊破碎。
    mask = median_filter(mask)
    # 8. 以 0 為門檻，將前景遮罩二值化。
    img_tresh = imgTreshold(mask)
    # 10. 根據二值遮罩分別取前景(fg)與背景(bg)的清晰像素，組成景深擴增影像。
    img_tresh_index = np.argwhere(img_tresh > 0)
    new_img = bgimg.copy()
    new_img[img_tresh_index[:, 0], img_tresh_index[:, 1]] = fgimg[img_tresh_index[:, 0], img_tresh_index[:, 1]]
    # 12. 儲存景深擴增影像。
    fg_hipass = cvtDtype(fg_hipass)
    bg_hipass = cvtDtype(bg_hipass)
    img_tresh = cvtDtype(img_tresh)
    img_tresh = cv2.cvtColor(img_tresh, cv2.COLOR_GRAY2BGR)
    print(img_tresh.shape)


    cv2.imwrite("hw1/fg_hipass.jpg", fg_hipass)
    cv2.imwrite("hw1/bg_hipass.jpg", bg_hipass)
    cv2.imwrite("hw1/img_tresh.jpg", img_tresh)
    cv2.imwrite("hw1/new_img.jpg", new_img)
    # cv2.imshow("a", new_img)
    # cv2.waitKey(0)
    # cv2.destroyAllWindows
```

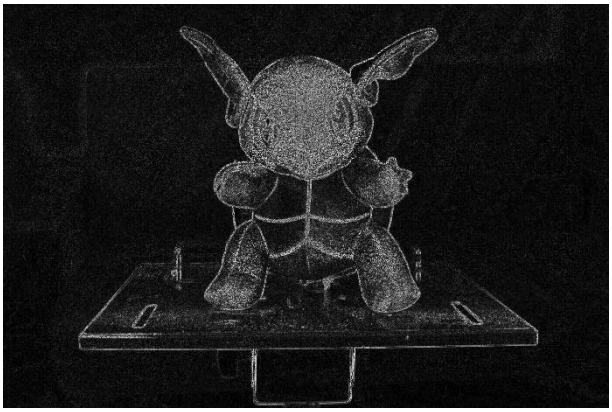

Figure 1. 對焦在前景的圖片套用 Laplacian 濾鏡

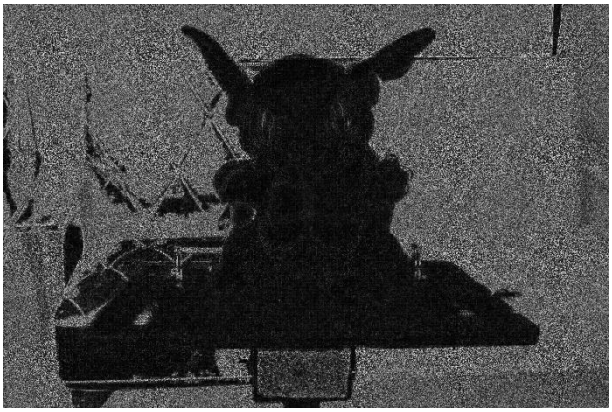

Figure 2. 對焦在後景圖片套用 Laplacian 濾鏡

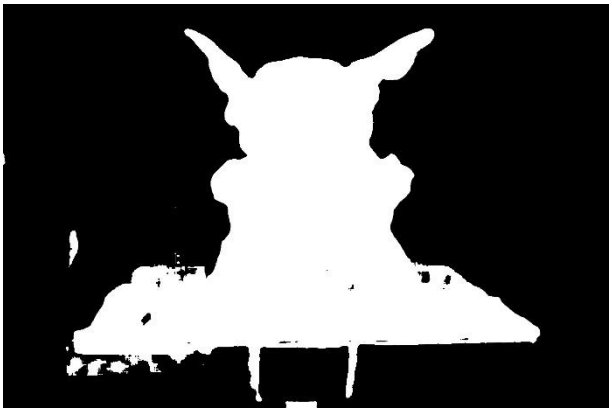

Figure 3. 二值化後的情景遮罩

Figure 4. 景深擴增影像

第二部分「視覺異常模擬」

```
import cv2
import numpy as np

M = np.array([[0.412453, 0.357580, 0.180423],  # Matrix of RGB to XYZ
              [0.212671, 0.715160, 0.072169],
              [0.019334, 0.119193, 0.950227]])


def __f__(img_XYZ):
    return np.power(img_XYZ, 1 / 3) if img_XYZ > 0.00856 else (7.787 * img_XYZ) /
(16 / 116)


def __anti_f__(img_XYZ):
    return np.power(img_XYZ, 3) if img_XYZ > (6 / 29) else 3 * ((6 / 29) ** 2) *
(img_XYZ - 4 / 29)


# region of RGB to Lab
def __BGR2XYZ__(pixel):
    b, g, r = pixel[0], pixel[1], pixel[2]
    rgb = np.array([r, g, b])  # The order list is BGR via opencv. So I transform
the order form BGR to RBG
    XYZ = np.dot(M, rgb.T)
    XYZ = XYZ / 255  # normalize
    return XYZ[0] / 0.950456, XYZ[1] / 1, XYZ[2] / 1.088754


def __XYZ2Lab__(XYZ):
    F_XYZ = [__f__(X) for X in XYZ]
    L = 116 * F_XYZ[1] - 16 if XYZ[1] > 0.00856 else 903.3 * XYZ[1]
    a = 500 * (F_XYZ[0] - F_XYZ[1])
    b = 200 * (F_XYZ[1] - F_XYZ[2])
    return L, a, b


def BGR2Lab(img):
    h = img.shape[0]
    w = img.shape[1]
```

```
    Lab = np.zeros([h, w, 3])
    for y in range(h):
        for x in range(w):
            XYZ = __BGR2XYZ__(img[y, x])
            result = __XYZ2Lab__(XYZ)
            Lab[y, x] = result[0], result[1], result[2]
    return Lab
# end region

# region of Lab to BGR
def __Lab2XYZ__(Lab):
    fY = (Lab[0] + 16.0) / 116.0
    fX = (Lab[1] / 500.0) + fY
    fZ = fY - Lab[2] / 200.0

    X = __anti_f__(fX)
    Y = __anti_f__(fY)
    Z = __anti_f__(fZ)

    X = X * 0.95047
    Y = Y * 1
    Z = Z * 1.0883
    return X, Y, Z


def __XYZ2RGB__(XYZ):
    XYZ = np.array(XYZ)
    XYZ = XYZ * 255
    rgb = np.dot(np.linalg.inv(M), XYZ.T)
    rgb = np.uint8(np.clip(rgb, 0, 255))
    return rgb


def Lab2BGR(img):
    h = img.shape[0]
    w = img.shape[1]
    new_img = np.zeros([h, w, 3])
    for y in range(h):
        for x in range(w):
```

```
            XYZ = __Lab2XYZ__(img[y, x])
            RGB = __XYZ2RGB__(XYZ)
            new_img[y, x] = RGB[2], RGB[1], RGB[0]
    new_img = new_img.astype(np.uint8)
    return new_img
# end region

def rg_blind(Lab):
    h = Lab.shape[0]
    w = Lab.shape[1]
    rg_img = np.zeros([h, w, 3])
    for y in range(h):
        for x in range(w):
            pixel = Lab[y, x]
            if pixel[1] != 0:
                pixel[1] = 0
            else:
                pixel[1] = pixel[1]
            rg_img[y, x] = (pixel[0], pixel[1], pixel[2])
    return rg_img


def yb_blind(Lab):
    h = Lab.shape[0]
    w = Lab.shape[1]
    yb_img = np.zeros([h, w, 3])
    for y in range(h):
        for x in range(w):
            pixel = Lab[y, x]
            if pixel[2] != 0:
                pixel[2] = 0
            else:
                pixel[2] = pixel[2]
            yb_img[y, x] = (pixel[0], pixel[1], pixel[2])
    return yb_img


def matlab_style_gauss2D(shape=(5, 5), sigma=0.5):
    """
```

```
    2D gaussian mask - should give the same result as MATLAB's
    fspecial('gaussian',[shape],[sigma])
    """
    m, n = [(ss - 1.) / 2. for ss in shape]
    y, x = np.ogrid[-m:m + 1, -n:n + 1]
    h = np.exp(-(x * x + y * y) / (2. * sigma * sigma))
    h[h < np.finfo(h.dtype).eps * h.max()] = 0
    sumh = h.sum()
    if sumh != 0:
        h /= sumh
    return h


def glaucoma(img):
    h = img.shape[0]
    w = img.shape[1]
    fil = matlab_style_gauss2D(shape=(h, w), sigma=100)
    fil = fil / np.nanmax(fil)
    b, g, r = cv2.split(img)
    b = np.multiply(b, fil)
    g = np.multiply(g, fil)
    r = np.multiply(r, fil)
    result = cv2.merge([b, g, r])
    result = result.astype(np.uint8)
    return result


def rgBlindSim(img):
    lab = BGR2Lab(img)
    rg = rg_blind(lab)
    new_img = Lab2BGR(rg)
    return new_img


def ybBlindSim(img):
    lab = BGR2Lab(img)
    yb = yb_blind(lab)
    new_img = Lab2BGR(yb)
    return new_img
```

```
if __name__ == "__main__":
    img = cv2.imread('cry1.jpg')
    # 1. 紅綠色盲：自行找一張色彩豐富的圖片，將 RGB 影像轉換至浮點格式，再轉換至 LAB 空間，將 a*設為 0，再轉回 RGB 空間。
    rg_blind_img = rgBlindSim(img)
    # 2. 黃藍色盲：將 RGB 影像轉換至浮點格式，再轉換至 LAB 空間，將 b*設為 0，再轉回 RGB 空間。
    yb_blind_img = ybBlindSim(img)
    # 3. 青光眼：讀取 RGB 影像的尺寸，利用 fspecial 函式建立與影像同尺寸的 2D 高斯濾鏡 (Gaussain filter)，sigma 值必須很高，才有效果。
    #    將濾鏡數值矩陣的每個數值除以其最大值。再將濾鏡點對點乘上影像的 RGB 值。模擬青光眼患者視野狹窄的現象。
    gl_img = glaucoma(img)

    cv2.imwrite("hw2/rg_blind_img.jpg", rg_blind_img)
    cv2.imwrite("hw2/yb_blind_img.jpg", yb_blind_img)
    cv2.imwrite("hw2/gl_img.jpg", gl_img)

    # cv2.imshow("a", img)
    # cv2.imshow("b", rg_blind_img)
    # cv2.imshow("c", yb_blind_img)
    # cv2.imshow("d", gl_img)
    # cv2.waitKey(0)
    # cv2.destroyAllWindows
```

Figure 5. 一般人

Figure 6. 紅綠色盲

Figure 7. 黃藍色盲

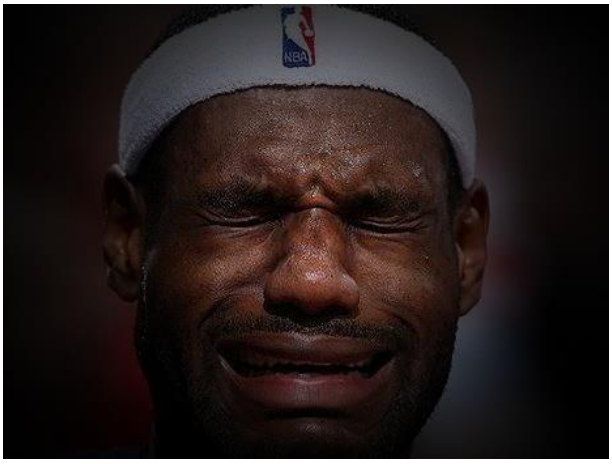

Figure 8. 青光眼

第三部分「以多維空間分析樹葉的差異」

```
import numpy as np
import cv2
import os


def imgMask(img, threshold):
    """
    :param img:
    :param threshold:
    :return: image mask in float32, [0, 1]
    """
    img_ = img.copy()
    h, w = img_.shape[0], img_.shape[1]
    output = np.zeros([h, w]).astype(np.float32)
    img_ = cv2.cvtColor(img_, cv2.COLOR_BGR2GRAY)
    img_ = img_.astype(np.float32)
    for y in range(img_.shape[0]):
        for x in range(img_.shape[1]):
            pixel = img_[y, x]
            if pixel > threshold:
                new_pixel = 0
            else:
                new_pixel = 1
            output[y, x] = new_pixel
    return output


def maskingImg(img, threshold):
    img_ = img.copy()
    img_mask = imgMask(img_, threshold)
    img_mask = np.argwhere(img_mask != 1)
    img_[img_mask[:, 0], img_mask[:, 1]] = 0
    img_ = (img_ / 255).astype(np.float32)
    return img_


def imgCenter(img):
    img_ = img.copy()
```

```
    img_ = imgMask(img_, 250)
    h, w = img_.shape[0], img_.shape[1]
    img_area = img_.sum()
    sum_x = 0
    sum_y = 0
    for y in range(h):
        for x in range(w):
            if img_[y, x] == 1.0:
                sum_y += y
                sum_x += x
    cy = round(sum_y / img_area)
    cx = round(sum_x / img_area)
    return cy, cx


def plotCenter(img):
    img_ = img.copy()
    h, w = img_.shape[0], img_.shape[1]
    cy, cx = imgCenter(img_)
    img_masked = imgMask(img_, 250)
    img_masked = cv2.cvtColor(img_masked, cv2.COLOR_GRAY2BGR)  # 3-D
    new_img = cv2.line(img_masked, (0, cy), (w, cy), (0, 0, 1), 1)
    new_img = cv2.line(new_img, (cx, 0), (cx, h), (0, 0, 1), 1)
    return new_img


def sigCurve(img, bins=60):
    img_ = img.copy()
    img_masked = imgMask(img_, 250)
    interval = 360 / bins
    r_histogram = np.zeros([bins, 1])
    cy, cx = imgCenter(img_)
    for y in range(img_masked.shape[0]):
        for x in range(img_masked.shape[1]):
            if img_masked[y, x] == 1:
                theta = np.mod(np.arctan2(cy - y, cx - x) * 180 / np.pi, 360)
                i = int(np.ceil(theta / interval))
                i = int(np.floor(theta / interval))
                r = np.sqrt((cy - y) ** 2 + (cx - x) ** 2)
```

```
                if r > r_histogram[i, 0]:
                    r_histogram[i] = r
    # x = np.arange(r_histogram.size)
    img_gradient = abs(np.gradient(r_histogram, axis=0)).sum()
    avg_img_gradient = img_gradient / bins
    return avg_img_gradient


def avgLightness(img):
    img_ = img.copy()
    img_masked = maskingImg(img_, 250)
    img_masked = cv2.cvtColor(img_masked, cv2.COLOR_BGR2GRAY).astype(np.float32)
    gray_index = np.argwhere(img_masked != 0)
    avg = img_masked.sum() / gray_index.shape[0]
    return avg


def redRatio(img):
    img_ = img.copy()
    img_masked = maskingImg(img_, 250)
    ch_ratio = []
    for ch in img_masked[:, :, 0], img_masked[:, :, 1], img_masked[:, :, 2]:  # BGR
        channel_index = np.argwhere(ch > 0)
        ch_avg = ch.sum() / channel_index.shape[0]
        ch_ratio.append(ch_avg)
    output = ch_ratio[2] / sum(ch_ratio)
    return output


def avgHiPass(img):
    img_ = img.copy()
    mask = imgMask(img_, 250)
    mask_indexing = np.argwhere(mask != 0)
    laplac_img = cv2.Laplacian(img_, -1, ksize=3)
    laplac_img = abs(laplac_img)
    output = laplac_img.sum() / mask_indexing.shape[0]
    return output


if __name__ == "__main__":
```

```
    y = 800
    x = 800
    blank_img_lr = np.ones([y, x, 3]).astype(np.float32)
    blank_img_ls = np.ones([y, x, 3]).astype(np.float32)

    folder_dir = "C:/Users/cghsi/Desktop/HW2/leaves/"
    list1 = []
    for i in os.listdir(folder_dir):
        img = cv2.imread(folder_dir + i)
        img_float = img.copy()
        img_float = img_float / 255
        masked_img = imgMask(img, 250)
        masked_img_index = np.argwhere(masked_img != 0)
        # 平均亮度
        img_avg_lightness = avgLightness(img)
        # 紅色平均占比
        img_red_ratio = redRatio(img)
        # 樹葉特徵分布圖 1 的座標
        by = np.floor(img_avg_lightness * y).astype(np.int32)
        bx = np.floor(img_red_ratio * x).astype(np.int32)
        # 平均高頻強度
        img_avg_lap = avgHiPass(img) / 400
        # 簽名曲線的梯度平均
        img_avg_sig = sigCurve(img) / 6.120549287353109
        # 樹葉特徵分布圖 2 的座標
        cy = np.floor(img_avg_lap * y).astype(np.int32)
        cx = np.floor(img_avg_sig * x).astype(np.int32)
        # 樹葉特徵分布圖 1
        blank_img_lr[masked_img_index[:, 0] + (y - by), masked_img_index[:, 1] + bx]
= img_float[masked_img_index[:, 0],

         masked_img_index[:, 1]]
        # 樹葉特徵分布圖 2
        blank_img_ls[masked_img_index[:, 0] + (y - cy), masked_img_index[:, 1] + cx]
= img_float[masked_img_index[:, 0],

         masked_img_index[:, 1]]
```

```
blank_img_ls = (blank_img_ls * 255).astype(np.uint8)
blank_img_lr = (blank_img_lr * 255).astype(np.uint8)

cv2.imwrite("hw3/blank_img_ls.jpg", blank_img_ls)
cv2.imwrite("hw3/blank_img_lr.jpg", blank_img_lr)
# cv2.imshow("a", blank_img_lr)
# cv2.imshow("b", blank_img_ls)
# cv2.waitKey(0)
# cv2.destroyAllWindows
```

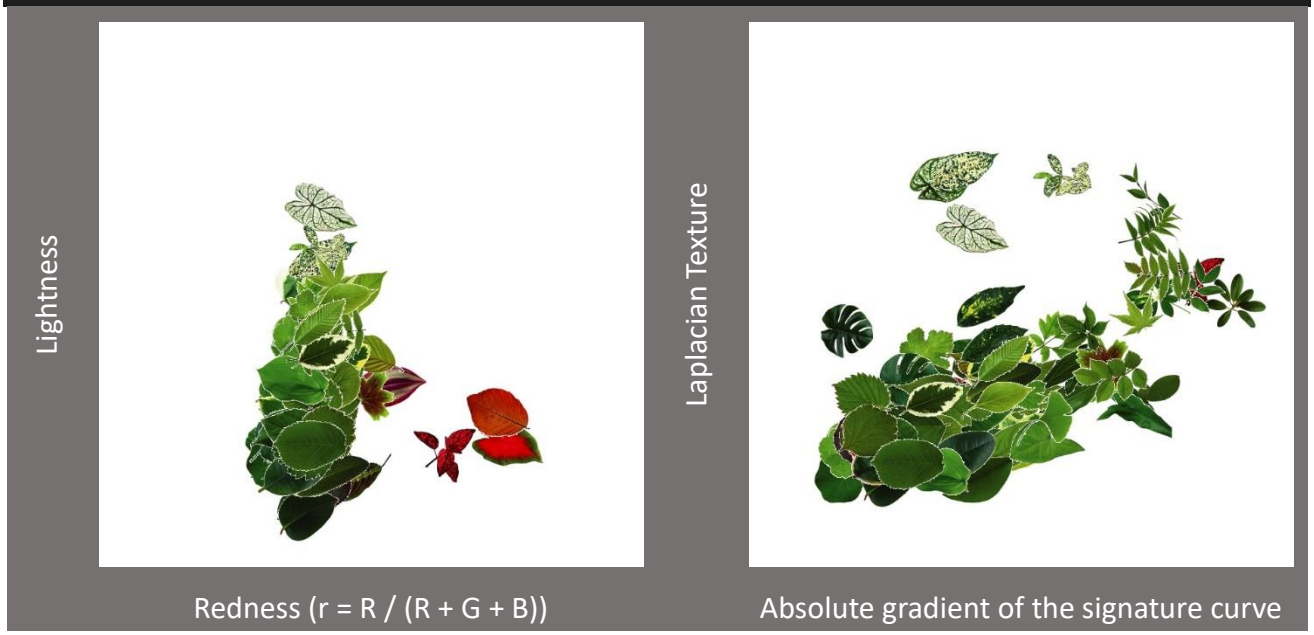

Figure 9. 多維樹葉特徵分析圖一